EEPROM™

Manual de usuario

Todos los sistemas de desarrollo de Mikroelektronika disponen de un gran número de módulos periféricos, ampliando el rango de aplicaciones de los microcontroladores y facilitando el proceso de probar los programas. Aparte de estos módulos, es posible utilizar un gran número de módulos adicionales conectados al sistema de desarrollo por los conectores de puertos de E/S. Algunos de estos módulos adicionales pueden funcionar como dispositivos autónomos sin estar conectados al microcontrolador.

Placa adicional

MikroElektronika
SOFTWARE AND HARDWARE SOLUTIONS FOR EMBEDDED WORLD ...making it simple

Placa adicional EEPROM

La placa *EEPROM* se utiliza como memoria EEPROM adicional utilizada para almacenar los datos que se deben guardar incluso en el caso de apagar la fuente de alimentación. La placa dispone de un chip de memoria 24C02 de 1Kbit que intercambia los datos con el microcontrolador por medio de la comunicación I2C. La placa EEPROM está conectada al puerto de E/S del microcontrolador pensado para la comunicación I2C por medio de un conector IDC de 10 pines. La presencia de la fuente de alimentación en la placa es indicada por un LED.

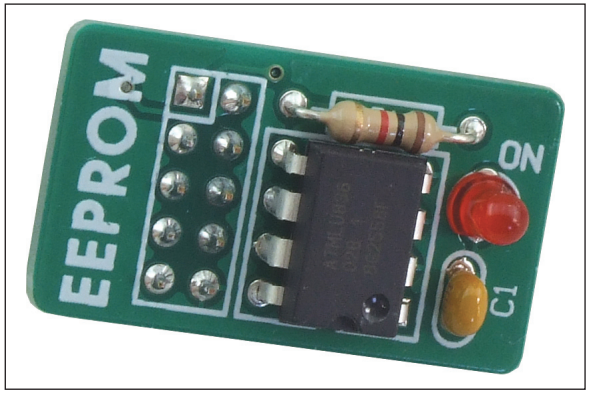


Figura 1: Placa adicional *EEPROM*

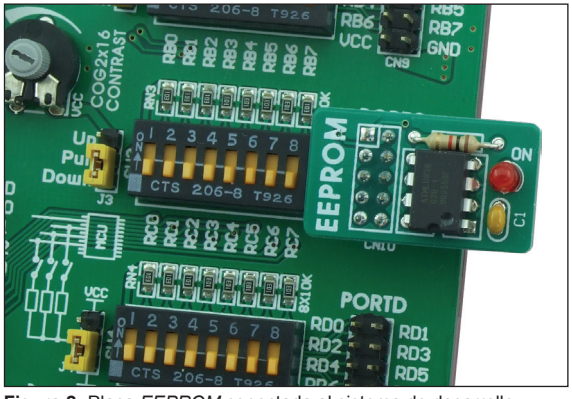


Figura 2: Placa *EEPROM* conectada al sistema de desarrollo

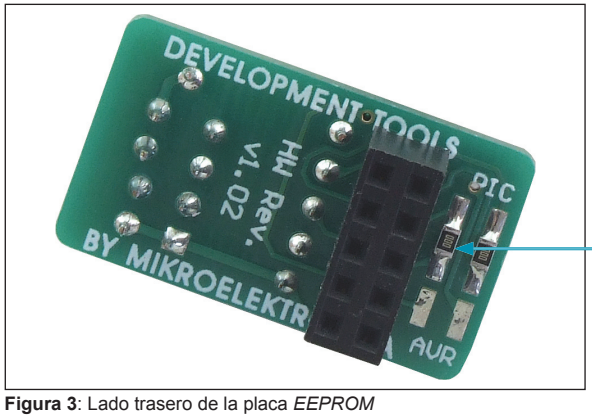


Figura 3: Lado trasero de la placa *EEPROM*

Los puentes en el fondo de la placa EEPROM se utilizan para seleccionar el microcontrolador que utilizará la memoria EEPROM.
Al poner los puentes en la posición PIC, la placa *EEPROM* se puede conectar a todos los sistemas de desarrollo de Mikroelektronika que soportan a los microcontroladores PIC.
Al poner los puentes en la posición AVR, la placa *EEPROM* se puede conectar a todos los sistemas de desarrollo de Mikroelektronika que soportan a los microcontroladores AVR.

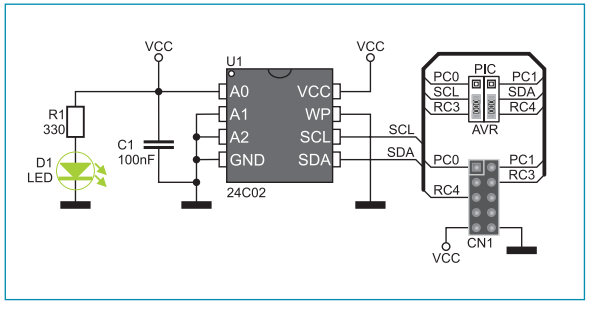


Figura 4: Esquema de conexión de la placa *EEPROM*

NOTA: En vez del chip de memoria mencionado anteriormente, Mikroelektronika puede incorporar otros chips de memoria de una capacidad más grande en las placas adicionales EEPROM. Aparte de esta característica, estas placas adicionales son las mismas que las placas EEPROM descritas aquí.

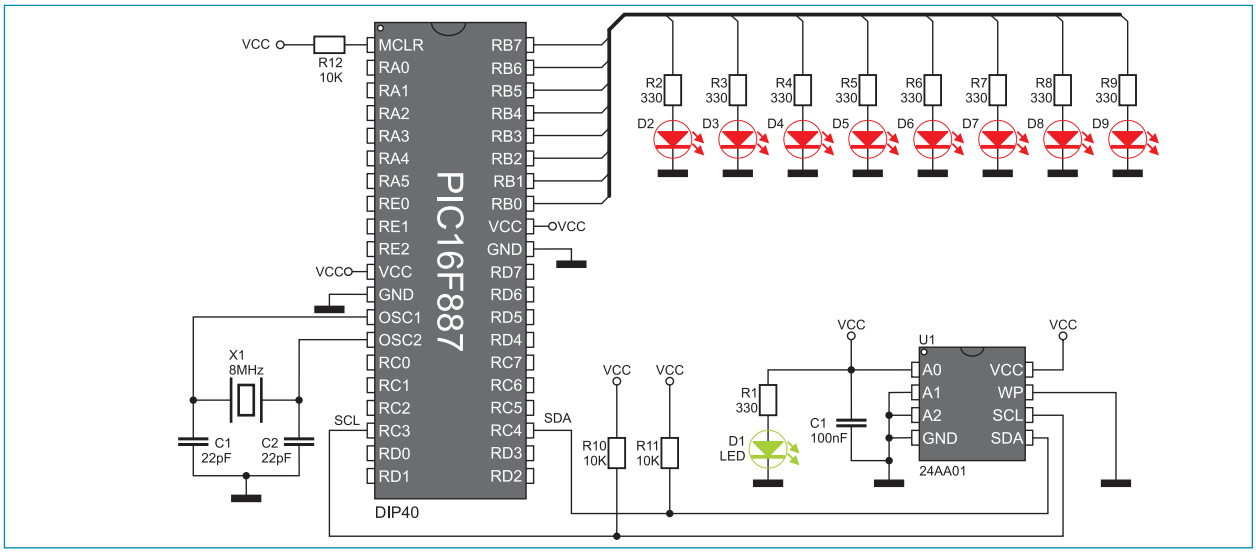


Figura 5: Esquema de conexión del módulo de memoria EEPROM adicional 24C02 al microcontrolador

Este simple ejemplo muestra cómo utilizar funciones para leer y escribir los datos de/en la memoria EEPROM adicional. Estas funciones se almacenan en las librerías de programa. Al establecer la comunicación I2C con el chip de memoria EEPROM 24C02, el microcontrolador escribe el dato en alguna localidad de memoria, luego lo lee y lo envía al puerto PORTB. El mismo programa está escrito en los lenguajes de programación mikroC, mikroBasic y mikroPascal. El microcontrolador PIC16F887 se utiliza en todos los ejemplos.

Ejemplo 1: Programa escrito en el compilador mikroC PRO for PIC

```
void main(){
  ANSEL  = 0;          // Configurar los pines AN como digitales de E/S
  ANSELH = 0;

  C1ON_bit = 0;        // Deshabilitar comparadores
  C2ON_bit = 0;

  PORTB = 0;
  TRISB = 0;           // Configurar el puerto PORTB como salida

  I2C1_Init(100000); // Inicializar la comunicación I2C
  I2C1_Start();        // Generar la señal de inicio I2C
  I2C1_Wr(0xA2)   // Enviar byte por medio de I2C (dirección del dispositivo + W)
  I2C1_Wr(2);        //  Enviar byte (dirección de la localidad en la EEPROM)
  I2C1_Wr(0xAA); //  Enviar datos (datos a ser escritos)
  I2C1_Stop();       //  Generar la señal de parada I2C

  Delay_100ms();

  I2C1_Start();       // Generar la señal de inicio I2C
  I2C1_Wr(0xA2);  // Enviar byte por medio de I2C (dirección del dispositivo + W)
  I2C1_Wr (2);         // Enviar byte (dirección de datos)
  I2C1_Repeated_Start(); // Generar la señal I2C para repetir el inicio
  I2C1_Wr(0xA3);             // Enviar byte (dirección del dispositivo + R)
  PORTB = I2C1_Rd(0u); // Leer los datos (NO reconocimiento)
  I2C1_Stop();                // Generar la señal de parada I2C
}
```

Ejemplo 2: Programa escrito en el compilador mikroBasic PRO for PIC

```
program I2C_Simple

main:
  ANSEL  = 0          ‘ Configurar los pines AN como digitales de E/S
  ANSELH = 0
  PORTB = 0
  TRISB = 0           ‘ Configurar el puerto PORTB como salida

  I2C1_Init(100000)              ‘ Inicializar la comunicación I2C
  I2C1_Start()                   ‘ Generar la señal de inicio I2C
  I2C1_Wr(0xA2)   ‘ Enviar byte por medio de I2C (dirección del dispositivo +W)
  I2C1_Wr(2)       ‘  Enviar byte (dirección de la localidad en la EEPROM)
  I2C1_Wr(0xAA) ‘  Enviar datos (datos a ser escritos)
  I2C1_Stop()      ‘  Generar la señal de parada I2C
  Delay_100ms()

  I2C1_Start()      ‘ Generar la señal de inicio I2C
  I2C1_Wr(0xA2)  ‘ Enviar byte por medio de I2C (dirección del dispositivo + W)
  I2C1_Wr(2)         ‘ Enviar byte (dirección de datos)
  I2C1_Repeated_Start()  ‘ Generar la señal I2C para repetir el inicio
  I2C1_Wr(0xA3)             ‘  Enviar byte (dirección del dispositivo + R)
  PORTB = I2C1_Rd(0)    ‘  Leer los datos (NO reconocimiento)
  I2C1_Stop()                ‘ Generar la señal de parada I2C
end.
```

Ejemplo 3: Programa escrito en el compilador mikroPascal PRO for PIC

```
program I2C_Simple;

 begin
  ANSEL  := 0;       // Configurar los pines AN como digitales de E/S
  ANSELH := 0;
  PORTB := 0;
  TRISB := 0;        // Configurar el puerto PORTB como salida

  I2C1_Init(100000);             // Inicializar la comunicación I2C
  I2C1_Start();                  // Generar la señal de inicio I2C
  I2C1_Wr(0xA2); // Enviar byte por medio de I2C (dirección del dispositivo +W)
I2C1_Wr(2);          // Enviar byte (dirección de la localidad en la EEPROM)
  I2C1_Wr(0xAA); //  Enviar datos (datos a ser escritos)
  I2C1_Stop();     // Generar la señal de parada I2C
  Delay_100ms();

  I2C1_Start();      // Generar la señal de inicio I2C
  I2C1_Wr(0xA2); // Enviar byte por medio de I2C (dirección del dispositivo + W)
  I2C1_Wr(2);       // Enviar byte (dirección de datos)
  I2C1_Repeated_Start();  // Generar la señal I2C para repetir el inicio
  I2C1_Wr(0xA3);            // Enviar byte (dirección del dispositivo + R)
  PORTB := I2C1_Rd(0);   // Leer los datos (NO reconocimiento)
  I2C1_Stop();              // Generar la señal de parada I2C
 end.
```